

Configuration Management for Web-based Instructional Systems

YANNIS PSAROMILIGKOS

Technological Education Institute of Piraeus
General Department of Mathematics
Computer Science Laboratory
Tel. ++3010 5381193, Fax. ++3010 5381351
E-mail: jpsa@teipir.gr

and

SYMEON RETALIS

University of Cyprus
Department of Computer Science
75 Kallipoleos Str., P.O. Box 20537CY-1678 Nicosia, Cyprus
Tel: +357-2-892246, Fax: +357-2-339062
Email: retal@cs.ucy.ac.cy

The design of web-based instructional systems is largely conducted on an intuitive, ad hoc basis, thus resulting in inefficient systems that do not capitalise on the strengths of networked technologies. Among these strengths, one could mention: multiple media delivery formats, synchronous and asynchronous communication and relaxation of constraints in place and time of instructional delivery. Moreover, the web-based instructional systems are inherited the growing problem hidden beneath the tremendous benefits of the World Wide Web (WWW) which is the management of the web's infrastructure and its content. Change is inevitable when computer software is built, but the web-based systems undergo changes more often and quite extensively, in their development and operational period. Additionally, a web-based instructional system must be developed through an evolution rather than a revolution, meaning that it is not released in one "big bang" at the end of the project. This is because the client's perception (either learner's or teacher's or institutional organisation's) of the complete end product is a "moving target". So, there is a need for a disciplined approach to managing the evolution of such systems. Such a disciplined approach that is applied throughout the software engineering process is called software configuration management (SCM). This paper begins a discussion of how the development of web-based instructional systems could benefit from SCM techniques and tools in order to achieve better learning effectiveness as well as development efficiency.

Categories and Subject Descriptors: D.2.9 **Software Engineering**: Management – *Software Configuration Management*; K.3.1 **Computers and Education**: Computer Uses in Education;
General Terms: Design, Experimentation, Human Factors, Management
Additional Key Words and Phrases: Web-based Instructional Systems, Learning Technology Systems, e learning, learning resources, Learning Management Systems, Instructional design, evaluation, version control

1. INTRODUCTION

The vision of a knowledge-based future where acquiring and acting on information and knowledge is the primary operation of all learners, is not very far from coming true [Sun 1999]. In order to facilitate the realization of this vision, Learning Technology Systems are being extensively employed. Learning Technology Systems (LTS) are learning, education and training systems that are supported by the Information Technology [IEEE LTSC 2001a]. Examples of such systems are computer-based training systems, intelligent tutoring systems and Web-based Instructional Systems. A special kind of LTSs are the Web-based Instructional Systems (WbIS) which are based on the state-of-the-art Internet and WWW technologies in order to provide education and training following the open and distance learning paradigm. WbIS aim to support and partially automate the

instructional process on a subject field, which might concern, for example, a course, a seminar or even a series of lectures [Gagné et al., 1994]. From a different perspective, these systems intend to satisfy certain instructional needs for a subject domain, which have surfaced mainly because of the advances in research and technology, the emergence of the information society and the globalisation of markets.

It is not odd, that WbIS, which make extensive use of new technologies, have been called upon to solve the vast demand for life-long learning, caused by the advances in technology itself. In particular, the advantages gained from employing Web-based Instructional Systems are derived from their potential to: promote advanced interactivity between learners and tutors; offer flexibility concerning the way, time and place of learning; grant multiple media delivery methods through hypermedia; allow several synchronous and asynchronous communication facilities; and provide easy, one-stop maintenance and reusability of resources [McCormack & Jones, 1997; Lowe & Hall 1999]. The design and implementation of such systems though is not an easy task, since they are complex systems that incorporate a variety of organizational, administrative, instructional and technological components [Moore & Kearsley 1996; Carlson, 1998]. A WbIS should be seen as a system comprised of three interrelated subsystems:

1. The human subsystem, which describes the roles, in as much detail as possible, for each kind of human agent (teacher, learner, tutor, network administrator) involved in the instructional process [Lindner, 2001].
2. The web-based learning resources subsystem, which is perceived as a mosaic of online learning resources. Such learning resources can be course notes, slideware, study guides, self-assessment questionnaires, communication archives, learning material used for communication purposes, etc.
3. The technical infrastructure subsystem, which is divided into common and special. An instructional system basically makes use of services from common infrastructure, which is a set of learning places, that support student learning in general (e.g. laboratories, networking facilities, etc.). However, in order to best support the instructional process, special infrastructure should be created (e.g. multimedia conferencing systems, state of the art hardware components, a specific learning management system, etc.), which will provide services unique to a particular instructional problem [Ford et al., 1996].

Systematic, disciplined development approaches must be devised in order to leverage the complexity and assortment of WbISs and achieve overall quality within specific time and fund limits. Additionally, the size and complexity of modern WbISs bring about great intricacy in their crafting as they evolve, as there is not enough knowledge or experience in this field. This also imposes the use of new disciplined approaches to managing their evolution. Such a disciplined approach can be adopted/adapted from the software engineering field.

Software Configuration Management (SCM) is the discipline of managing the evolution of software systems both during the initial stages of development and during all stages of maintenance. SCM constitutes a key element of the software engineering process and includes many activities that must be carried out consistently. Moreover, it involves many different individuals, such as customers, managers and software engineers as well as many different products such as management plans, specifications (requirements, design, test), code (source and executable), user's manuals, etc. SCM has four coordinating functions [IEEE 1987; IEEE 1990a; IEEE 1990b]

1. Configuration identification: The definition of the software life cycle products that will be under control, their baselines and how they will change.

2. Configuration Control: The technical and administrative procedures in order to control the changes to products.
3. Configuration Audit: The function that makes the current status of any software product visible to management.
4. Configuration Status Accounting: The function that provides the development history of any software product, recording the activities of the previous SCM functions.

In large-scale software systems the management of these activities is an extremely difficult work, but essential for effective and reliable evolution of such software. Computer assistant for the representation and evolution of the history of software system development is of great importance. It can avoid the confusion caused by interaction among the different individuals, improving productivity.

In this paper we try to illustrate the need to incorporate (via adaptation) SCM techniques to the development process of a WbIS.

The structure of the paper is as follows: Section 2 presents the SCM process that are being applied in general (i.e. non instructional) software systems. We should note that up to our knowledge, this is the first time that ideas on how to apply SCM functions to WbIS development are being written down. While section 3 describes the overall development process of a WbIS, section 4 will analyse the way that SCM can be incorporated into WbIS development process. Finally, concluding remarks and future plans will be mentioned in section 5.

2. THE SOFTWARE CONFIGURATION MANAGEMENT PROCESS

When a large number of people work on a software project, some of the problems that can arise are the following:

- Simultaneous update: When two or more programmers work separately on the same program, incompatibility of the changes they make can be disastrous.
- Shared or common code: When a change is made to a module shared by several subsystems, all the programmers involved in the development of these subsystems must be informed of the potential effects this change has on their work.
- Versions: Changes must be applied in a controlled manner if regression is to be avoided. Moreover, they result in a great number of versions even of a single component and therefore the structure of the system becomes extremely complex.

All the items that are produced as part of the software engineering process are collectively called a software configuration. SCM is the process of identifying, organizing and controlling changes to software configurations made in all stages of a software project's life cycle. Its purpose is to maximize the productivity of the development team members by controlling their interaction and minimizing mistakes.

Because change can occur at any time, SCM is an “umbrella” activity that is applied throughout the software engineering process. SCM activities provide means for:

- identifying configurations and changes
- controlling the application of changes
- ensuring the proper implementation of changes
- reporting changes to all the interested team members.

The output of the software engineering process is artefacts that can be divided into several classes. Moreover, we can view and manage these artefacts as composite or atomic entities. The term software configuration item (SCI) used to denote such a

controllable (under configuration management control) artefact or a part of it. If we view an item as a collection of other items, then it is called composite and atomic otherwise. The SCIs may form hierarchies according to the relation “is_composed_of” among composite and their parts (atomic or composite). A SCI can be any of the parts of a whole system as long as it is treated and managed as a single unit. Of course, management is simpler if the item belongs to the lower levels of the system's hierarchy.

Because changes are inevitable SCIs may have many versions. The versions of atomic SCIs imply the versions of composite SCIs and so on. Because of versions the structure of a software system becomes extremely complex. There are two kinds of versions: revisions and variations. Variations represent independent lines of development and each one consists of a sequence of revisions. Variations coexist in time while revisions replace each other because their purpose is primarily to fix bugs. When a SCI version is distributed outside the development organisation is called release. Version control of SCIs is one of the fundamental tasks of every software configuration management tool and it constitutes the basis of many research efforts till now [Estublier, 1988; Estublier, 1995; Estublier, 1999; Conradi, 1997; Magnusson, 1998; André van der Hoek, 2001].

Changes also need a key mechanism to bring them under control because they can rapidly lead to chaos. The key mechanism for managing changes is the baseline. A baseline is a milestone in the software engineering process that marks the completion of a phase accompanied by the delivery of a number of approved (validated) SCIs. A baseline is the foundation of configuration management as it provides the official standard on which subsequent work is based and therefore they should be established at an early development point. All the SCM processes revolve around the baselines. Once an initial product level has stabilised, a first baseline is established. Every successive set of validated enhancements establishes a new baseline that provides the cornerstone for further development.

Once a SCI becomes a baseline, it is placed in a project database (also called a project library or software repository). The baselines must be protected against unauthorised change while at the same time enabling the programmers to modify and test their code. This flexibility is accomplished by providing the programmers with private working copies of any part of a baseline. Thus they can try out any changes without interfering with the work of anyone else. When their work is ready to be incorporated into a new baseline, change management ensures that each change introduced is compatible with every other and also maintains system integrity.

3. THE DEVELOPMENT PROCESS OF A WbIS

Nowadays, instructional systems make extensive use of networked technologies. A consequence of this trend is that developers build complex instructional systems that incorporate a variety of organisational, administrative, instructional, and technological components [Carlson, 1998]. In this section we briefly describe CADMOS methodology (Web-based Courseware Development Methodology for Open Learning Systems) which is a development methodology for WbISs [Retalis, 1998].

CADMOS supports the incremental model for the development of a WbIS [Schash, 1990]. This is because the incremental model is iterative and is characterised in a manner that enables developers to construct increasingly more complete versions of the end product. Thus, according to CADMOS, a WbIS should be developed as a series of fully functional builds (working versions of the instructional system). A build satisfies the current set of the requirements of the product under development. The underlying essence of the incremental development is that the client's perception (either learner's or

teacher's or institutional organisation's) of the complete end product is a "moving target". While the builds are tested and summatively evaluated, the user's opinion regarding the characteristics of the system might change, resulting in changing the requirements and consequently the design and development of the future builds.

The instructional development should contain three main processes, each one subdivided into sub-processes as illustrated in Figure 1:

- The instructional problem solving process
- The instructional system construction process
- The instructional system utilisation process.

The aim of the problem solution finding sub-process is to construct a desired solution to a given instructional problem already defined by the situational evaluation sub-process. The solution to the instructional problem emerges from blending five interrelated sets of learning elements: the learning objectives, the didactic events, the syllabus, the assessment procedure and other issues like prerequisites, fees, technical constraints, and so forth. This solution is illustrated graphically in Figure 2 and is the main product of this process. It must be a well written, highly detailed document following specific guidelines on how to formulate the learning objectives, the syllabus, the didactic events, etcetera (for

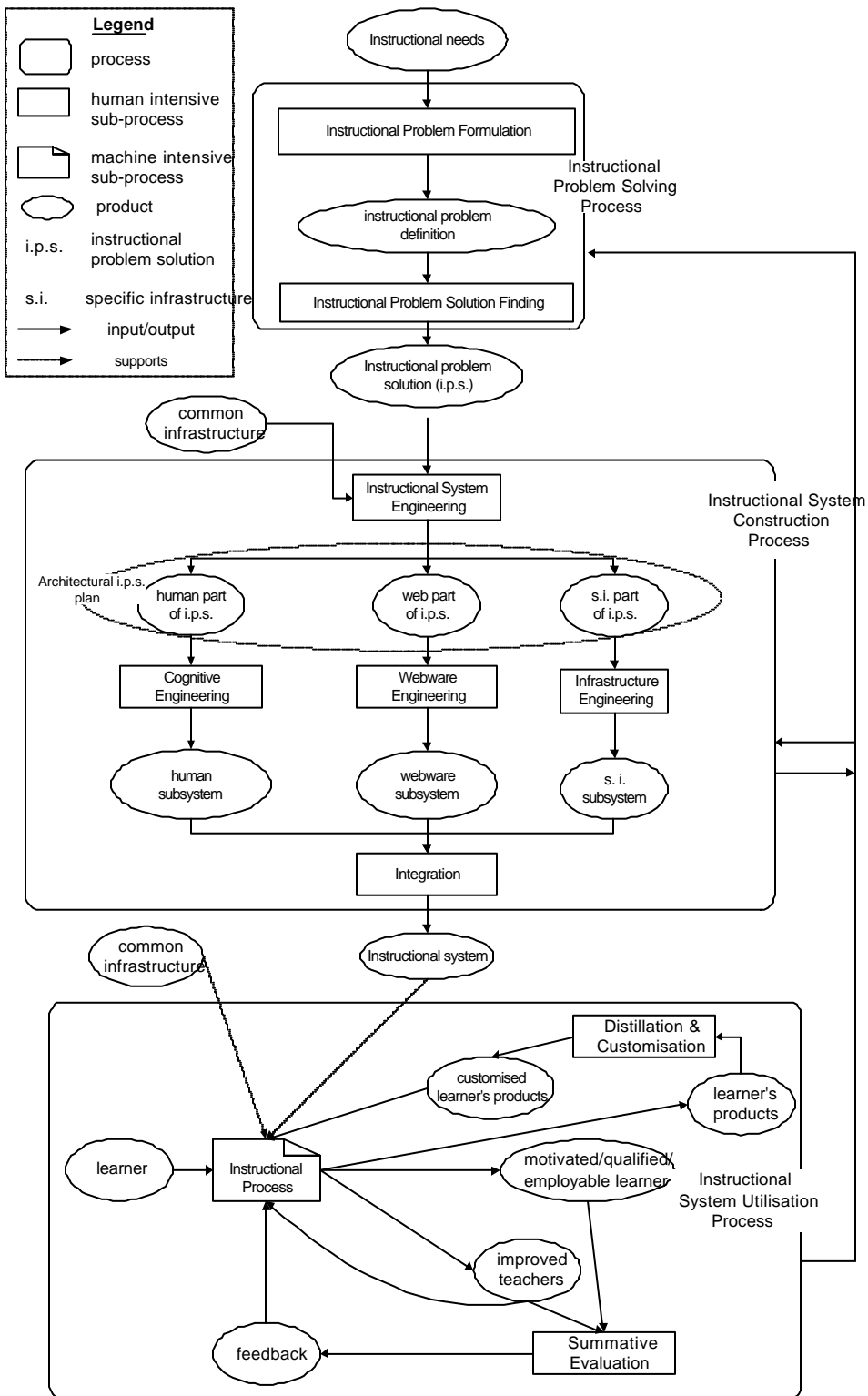


Fig. 1. Development stages of an WbIS according to CADMOS methodology.

an overview, see Gagné, Briggs, & Wager, 1994; Rowntree, 1994; Tennyson & Breuer, 1997). Thus, such a “non-technical” solution, that is the instructional problem abstract solution, plays the role of requirements specification for the WbIS under construction.

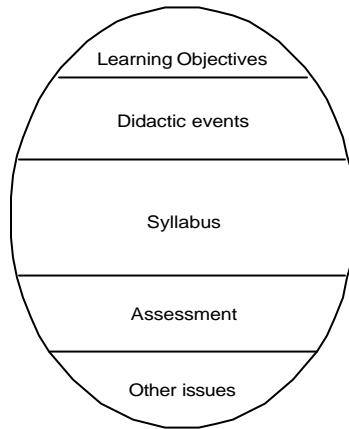


Fig. 2. An ideal solution of an instructional problem.

The construction process is the actual implementation of an instructional problem solution for a specific learning environment. The construction phase receives as input the solution from the problem finding process. Then, this solution is divided into three parts: (1) human part, (2) webware part, and (3) specific infrastructure part. This division is done within the systems engineering sub-process. Each part of the solution specifies how the solution will be realised and supported into a real learning environment. It should be noted that according to the holistic theory, each subsystem should be itself an independent, viable system. However, the systems are interrelated, thus we need to keep the whole in mind otherwise reductionism may distort our understanding of the overall. Having specified the solution parts, each one of them is developed according to related methodologies (i.e. cognitive engineering for training the human actors to play their roles, courseware engineering for developing the web-based learning resources and software engineering for building the specific software infrastructure). resulting the end, three subsystems that are all integrated into one “whole”.

The last stage of the process is concerned with the utilisation of an instructional system. It is within this process that the developed system will be summatively evaluated and reviewed. The utilisation process consists of:

- The instructional process when the instructional system developed is being used, tested and validated in a real learning environment.
- The summative evaluation of the instructional problem solution sub-process which provides feedback from both learners and teachers.
- The customization and distillation of the outputs of the instructional process which can be used for the evolution of the instructional system.

4. INCORPORATING SCM INTO WbIS DEVELOPMENT PROCESS

As shown in section 3, there is a number of interrelated subsystems (life cycles) involved in the development of a WbIS and so the configuration of the system becomes extremely

complex. A configuration management model should represent the development and evolution of WbIS configurations so that the various life cycles expressing the organisations' specific needs could be supported while at the same time it should ensure and maintain the consistency of the underlying configurations. In this section we try to define such a model for the representation of the WbIS configurations and their evolution.

At first, we consider a macro and a micro view of the WbIS development process in order to implement configuration management (Figure 3). The macro view involves the top-level processes (phases) of the WbIS development process, that is, "Problem Solving", "System Construction" and "System Utilisation" processes as well as the underlying output products. For the "System Utilisation" process there are only intermediate products that are used as input to other processes during evolution. At the micro view we deal with the analysis of the "System Construction" process because it involves three interrelated subsystems (life cycles) that produce the underlying complexity.

By looking through the above subsystems we can notice the following: The specific infrastructure subsystem that will provide services unique to the particular instructional problem is a software engineering task. So, the development organisation could follow current software configuration management techniques and tools for the development and evolution of such a system. As it concerns the human subsystem it is a task of the cognitive engineering field. However, when a configuration management process will be introduced in the life cycle of a WbIS system, new roles and competencies will be generated that needs to be identified and clearly specified. "Competency refers to a state of being well qualified to perform an activity, task or job function. When a person is competent to do something, he or she has achieved a state of competency that is recognisable and verifiable to a particular community of practitioners" [Spector and Teja 2001]. Typically, a competency is divided into specific indicators describing the requisite knowledge, skills, attitudes and context of performance [IBSTPI 2002; Richey et al. 2001; Goodyear et al. 2001].

The webware subsystem deals with the content of a WbIS system which inevitably concentrates the most interests and it needs further analysis. Moreover, Web content management is an open research topic both for configuration management area [Dart 1999; Murugesan et al., 2001] and Learning Technology Systems [IEEE LTSC 2001b]. Of course, a configuration management model should take into account the interrelations of all the subsystems in order to control the evolution of the WbIS system.

4.1 Modelling the WbIS system configurations

We model the WbIS system configurations as a three-dimensional space. The first dimension represents the configuration development, the second dimension represents the configuration evolution and the third dimension represents the decomposition of a configuration (Figure 4).

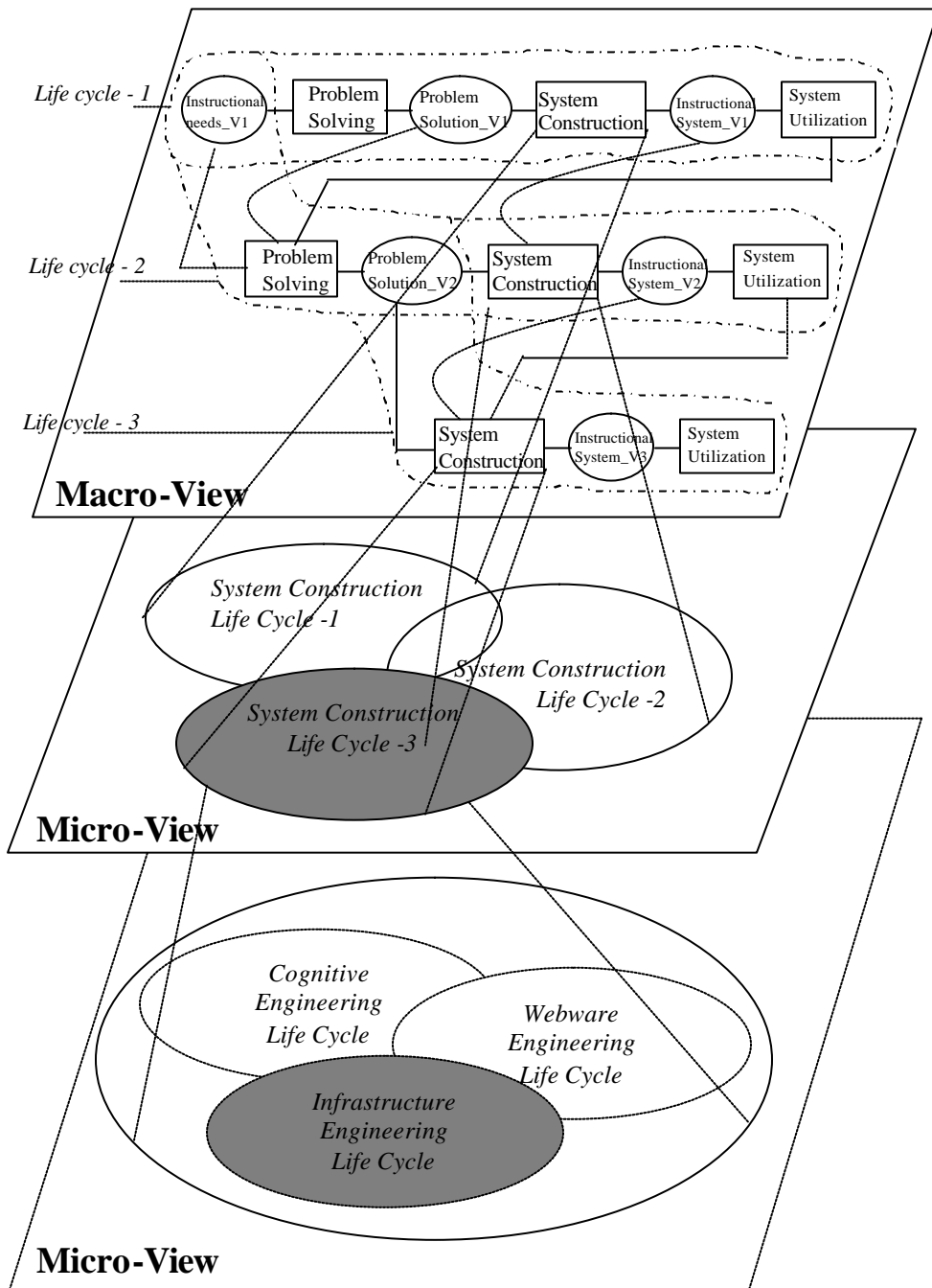


Fig. 3. Development and evolution of a WbIS.

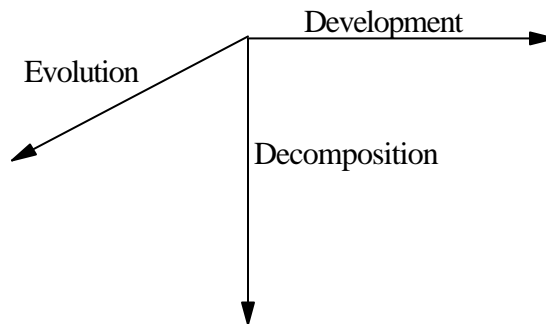


Fig. 4. A model for WbIS system configurations.

The dimension of development represents the way each configuration is developed. At the macro view each configuration involves the top-level phases of the WbIS system development process as shown in figure 1. Such a configuration defines a concrete version of the WbIS system. The configurations of the micro view represent how the top-level phases of the WbIS system development process are implemented.

The underlying micro views are subject of the decomposition process. During this process we analyse how each composite configuration management entity is constructed. If such an entity is composed of other composites we continue until we define all the elementary (atomic) items. We recall that one of the most important configuration management processes is the identification of the entities that will be under control, i.e. the identification of configuration management items.

The dimension of evolution represents how each configuration evolves over the time. We must note that each configuration once it is created and placed under configuration control (baseline) becomes immutable; that is, we must create a new version of it each time we have to change it. We consider the evolution process as actually being selective iterations of the development process. Such iterations at the macro view generate the versions of the WbIS system, while at the micro view generate the versions of the various products (configuration management items) comprising the WbIS system. Each evolution cycle normally begins after the system utilisation process when the WbIS system is summaratively evaluated. However, a formative evaluation process may initiate intermediate evolution cycles (i.e. evolution cycles at the micro view) before the start-up of the utilisation process.

4.2 An example

In this section we describe a WbIS system that has been developed and evolved at the National Technical University of Athens (NTUA), Greece. The system supports an introductory course in software engineering offered by the Software Engineering Laboratory of the Electrical and Computer Engineering Department at the NTUA. The first version of the WbIS system was evaluated in 1997 [Makrakis et al., 1998]. At the macro view the underlying version can be represented by the development process labelled as «Life cycle-1» in figure 3, while at the micro view, the structure of this version of the WbIS system is shown in figure 5. The cornerstone of the special technological infrastructure was the WebCT course management system [WebCT 2001]. This system hosted the web-based learning resources, the details about students and

instructors (personal data and records), and the data used for administration (course management).

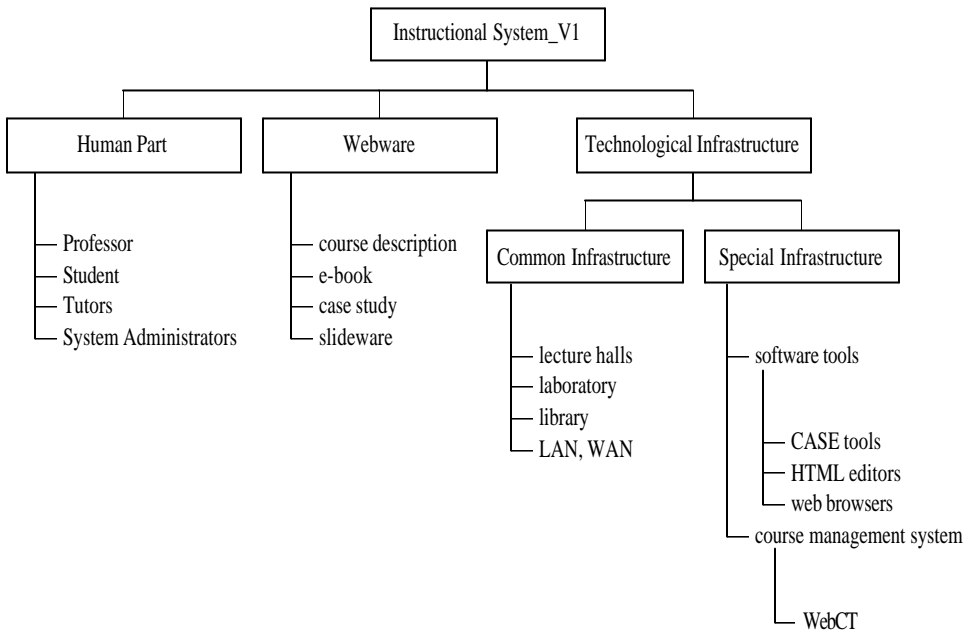


Fig. 5. The micro view of the first version of the WbIS system.

After its first evaluation a new development process started (i.e. an evolution cycle), in order to implement the necessary changes as suggested by the results from quantitative and qualitative analysis of students' feedback. For example, the evaluation study of the first version of the WbIS showed that the students were concerned with the following:

- There were not enough elaborated case studies dealing with software development.
- The course content did not include enough modern topics and open research issues.
- The learning material units were quite long and lacked multimedia for illustrating appropriate topics.
- Web-based interaction, and subjects for discussions were limited.

Thus, the next version of the system differed from the first one in the following ways:

- The Web-based learning material (webware) was enriched by three detailed case studies on software development using various methodologies.
- An on-line library with links to special and modern topics on software engineering was created.
- Serious cuts and changes in the context of the Web-based modules were made.
- The exam papers of the previous years with their solutions were added as learning resources.
- The subject matter experts provided some interesting discussion topics. The instructor and the tutors were better trained to moderate the discussion on these topics as well as to encourage and motivate students participating in the discussions.
- A new version of the WebCT system (versions 3.1) was installed which had better usability and functionality.

So, at the macro view a new version of the problem solution was created because the new version included two new didactic events (see figure 2) (synchronous

communication and self-assessment exercises). Therefore, at the macro view, the new version can be represented by the development process labelled as «Life cycle-2» in figure 3. At the micro view, the structure of this version of the WbIS system is shown in figure 6. The items in bold represent both new versions of previous items or new ones. For the human part it means, for example, that professor, tutors, and system administrators were more competent to play their roles or there were modifications to their roles. The second version of the WbIS system was evaluated in 2000 [Retalis, S., Psaromiligkos, Y., Anastasiades, A., 2002] and according to that study the corrective actions made based on the feedback from the first evaluation study did affect positively the learning effectiveness of the revised system.

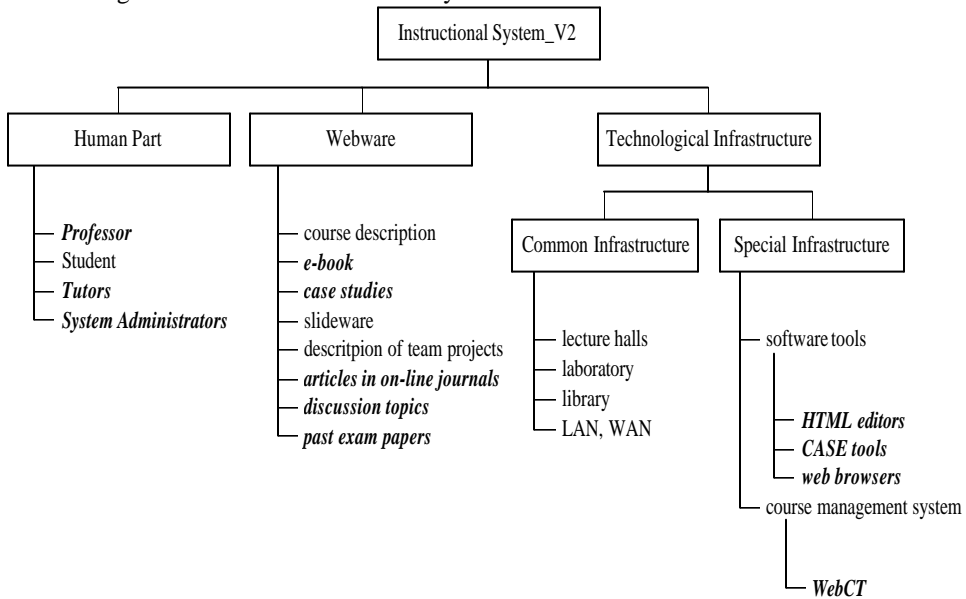


Fig. 6. The structure of the second version of the WbIS system.

The third version of the WbIS system included minor changes, which mainly concerned the incorporation of customised learner’s products from the corresponding utilisation process into the webware subsystem. For example, the best of the students’ team projects (software systems with their documentation) were included into the webware as specimen of team projects for the new coming students. Moreover, newer versions of the WebCT and the software tools had been installed. So, at the macro view the new version can be represented by the development process labelled as «Life cycle-3» in figure 3, while at the micro view the structure is similar to that of figure 6 with some modifications of the webware part.

Thus, the evolution history of the WbIS system for the introductory course in software engineering counts three versions till now, which can be characterised as revisions according to configuration management terminology. We must note the common parts of the configurations between subsequent versions denoting the reusability of resources as well as the need for version management. For example, the transition of content between the subsequent versions of the WebCT environment was done by using the backup/restore features of the system which is a very primitive function.

5. CONCLUDING REMARKS AND FEATURE PLANS

There is a need for a disciplined approach to managing the evolution of WbIS systems because they are complex systems, have long lifetime, and their representation and evolution history is very complex. This is a problem analogous to software engineering community for software systems and configuration management has the potential to provide a solution in this area [Dart, S., 1999; Murugesan et al., 2001].

In this paper we described a model for the development and evolution of WbIS system configurations. Our model couples the development and evolution process into one framework, which seems promising for the implementation of a configuration management process. This approach is not a formal method and it needs to mature in order to have computerised support.

So, among our future plans are the following:

- The refinement of the model at the micro views in order to formalise the configuration management process.
- The definition of a version model for the WbIS that will support both revisions and variations. In this work we mainly focused on the revision aspect of WbIS. However, we could have variations of the WbIS in order to support multilinguality and adaptability. We have already developed the first version of an adaptive WbIS and its underline architectural model [Papasalouros, A., and Retalis, S., 2002] and we will now work on applying the proposed model for the evolution of the system's configurations.

Finally, we are in the process of designing a configuration management tool for having a computer assistant in this process. We do not try from scratch but we try to make additions to existing tools that are being used in software configuration management. We also tend to persuade ourselves that both course management tools (like WebCT, Blackboard, etc.) and the learning objects management tools (which follow the LTSC LOM or IMS or EML standard) should have features that support versions of the specific infrastructure, the learning material and the WbIS in general.

REFERENCES

- ANDRE VAN DER HOEK. 2001. Software Configuration Management: New Practices, New Challenges, and New Boundaries. *Proceedings of the 10th International workshop on Software Configuration Management*, 23rd ICSE, Toronto, Canada, May 2001.
- CARLSON, P.A. 1998. Advanced Educational Technologies – Promise and Puzzlement. *Journal of Universal Computer Science* 4(3).
- CONRADI R. 1997. Software Configuration Management. *Proceedings of the 7th International workshop on Software Configuration Management*, ICSE'97, Volume 1235 of Lecture Notes in Computer Science, Boston, MA, USA, 1997, Springer-Verlang.
- DART S. 1999. Content Change Management: Problems for Web Systems. In *Proceedings of 9th International Symposium*, SCM-9, Toulouse, France, 1999, J. ESTUBLIER, Eds, Volume 1675 of Lecture Notes in Computer Science, 1-16, Springer-Verlang.
- ESTUBLIER J. 1988. Configuration Management: the notion and the tools. *Proceedings International Workshop on Software Version and Configuration Control* Stuttgart, FRG, 1988.
- ESTUBLIER J. 1995. Software Configuration Management. Selected papers from *International WorkShops on Software Configuration Management* SCM-4 and SCM-5, Volume 1005 of Lecture Notes in Computer Science, Seattle Washington, 1995, Springer-Verlang.

ESTUBLIER J. 1999. System Configuration Management. *Proceedings of the 9th International Symposium on Software Configuration Management, SCM-9*, Toulouse, France, 1999, Volume 1675 of Lecture Notes in Computer Science, Springer-Verlang.

FORD, P., GOODYEAR, P., HESELTINE, R., LEWIS, R., DARBY, J., GRAVES, J., SARTORIUS, P., HARWOOD, D. & KING. T. 1996. *Managing Change in Higher Education: A Learning Environment Architecture*, Open University Press, London.

GAGNE, R., BRIGGS, L., WAGER. L. 1994. *Principles of Instructional Design*. Fort Worth, TX: HBJ College Publishers.

GOODYEAR, P., SALMON, G., SPECTOR, M., STEEPLES, C., & TICKNER, S. 2001. Competencies for online teaching. *Educational Technology Research & Development* 49 (1), 65-72.

IBSTPI 2002. International Board of Standards for Training, Performance and Instruction (IBSTPI), <http://www.ibstpi.org/>.

IEEE 1987. IEEE/ANSI. IEEE Guide to Software Configuration Management. ANSI/IEEE Std 1042-1987. IEEE Press, New York, NY, USA.

IEEE 1990a. IEEE/ANSI. IEEE Standard for Software Configuration Management Plans. IEEE Std 828-1990. IEEE Press, New York, NY, USA.

IEEE 1990b. IEEE/ANSI. IEEE Standard glossary of software engineering terminology. IEEE Std 610.12-1990. IEEE Press, New York, NY, USA.

IEEE LTSC 2001a. IEEE Learning Technology Standards Committee (LTSC), Draft Standard for Learning Technology - Public and Private Information (PAPI) for Learners (PAPI Learner) — Core Features, Draft 8, November 2001, <http://ltsc.ieee.org>.

IEEE LTSC 2001b. IEEE Learning Technology Standards Committee, (LTSC), Draft Standard for Learning Object Metadata (LOM), Draft 6.4, 2001, <http://ltsc.ieee.org>.

LINDNER, R. 2001. Expertise and Role Identification for Learning Environments (ERILE), Proposed Standard Draft for German DIN NI-36, (http://www.igd.fhg.de/~lindner/PROMETEUS/SIG-DESIGN_Meeting-Point.htm)

LOWE, D. AND HALL, W. 1999. *Hypermedia & the Web: An Engineering Approach*, John Wiley Ltd.

MAGNUSSON, B. 1998. System Configuration Management. ECOOP'98 SCM-8 Symposium, Volume 1439 of Lecture Notes in Computer Science, Brussels, Belgium, Springer-Verlang.

MAKRAKIS, V., RETALIS, S., KOUTOUMANOS, A., PAPASPYROU, N., SKORDALAKIS, M. 1998. Evaluating the effectiveness of an ODL Hypermedia System and Courseware at the National Technical University of Athens: A Case Study. *Journal for Universal Computer Science*, 4(3), 259-272.

MCCORMACK, C. AND JONES, J. D. 1997. *Building a Web-based Education System*, Wiley Computer Publishing.

MOORE, M. G., AND KEARSLEY, G. 1996. *Distance Education: A Systems View*, Wadsworth Publishing Company.

MURUGESAN S., DESHPANDE Y., HANSEN S. and GINIGE A. 2001. Web Engineering: A New Discipline for Development of Web-Based Systems. In *Proceedings of the International Conference on WebEngineering 2000*, S. Murugesan and Y. Deshpande (Eds): Web Engineering 2000, Volume 2016 of Lecture Notes in Computer Science, 3-13, Springer-Verlang.

PAPASALOUIROS, A., RETALIS, S., 2002. Ob-AHEM: A UML-enabled model for Adaptive Educational Hypermedia Applications. *Interactive educational Multimedia*, ISSN 1576-4990, special issue on the theme "adaptive educational multimedia" (to appear).

RETALIS, S., 1998. *CADMOS: An Instructional Systems development methodology with emphasis on the construction of web-based learning resources*, PhD thesis, National Technical University of Athens (in Greek).

RETALIS, S., PSAROMILIGKOS, Y., ANASTASIADIS, A. 2002. The “why”, “what”, “when” and “how” of a summative evaluation method about the learning effectiveness of web-based learning systems”. *Themes in Education* (accepted for publication).

RICHEY, R. C., FIELDS, D. C., FOXON, M., ROBERTS, R. C., SPANNAUS, T. AND SPECTOR, J. M. 2001. *Instructional design competencies: The standards (3rd ed.)*. Syracuse, NY: ERIC Clearinghouse on Information and Technology.

SCHASH, S. R. 1990. Chapter 3: Software Life Cycle Models, In *Software Engineering*, Homewood, IL: Aksen Associates, 20-40.

SPECTOR, M. J., DE LA TEJA, I. 2001. *Competencies for Online Teaching*, NY: ERIC Clearinghouse on Information & Technology, EDO-IT-2001-09, December 2001 [<http://www.ericit.org/digests/EDO-IR-2001-09.shtml>].

SUN 1999. Sun Microsystems, Understanding Distance Learning Architectures, A White Paper, 1999.

WebCT. 2001. WebCT. [Online] Available: <http://www.webct.com/>